

Boolean model

- Popis projektu a cíle
- Způsoby řešení
- Implementace
- Příklad výstupu
- Experimentální sekce
- Diskuze
- Závěr

Autor: Volha Chukava (chukavol@fit.cvut.cz)

[Odkáz na git repozitář](#)

Popis projektu a cíle

Cílem projektu je vytvořit FullStack aplikací pro efektivnější vyhledávání textových výrazů v velkých kolekcích dokumentů v porovnání s lineárním vyhledáváním.

A konkrétně implementovat dva způsoby vyhledávání:

1. Vyhledávání pomocí boolean modelu - ukládání invertovaného seznamu.
2. A basic sekvenční průchod - $O(n)$

Cílem implementaci těchto dvou způsobů je porovnání jejich efektivnosti(rychlosti) při procházení datových kolekcí různé velikosti.

Vstup: Vstupem by měl být jakýkoliv textový boolean dotaz. Program by měl umět zpracovat i dlouhé výrazy včetně uzavorkování. Program by měl umět přijímat dotazy z GUI.

Výstup: Výstupem by měl být seznam všech dokumentů z kolekcí, které odpovídají zadanému dotazu. Program by měl umět zobrazit výsledky dotázování na GUI. Program by taky měl umět ukázat výhody použití boolean modelu - například porovnáním doby vyhodnocení výrazů způsoby 1 a 2.

Způsoby řešení

1. Extrakce a preprocessing termů z dokumentů.

Extrakce a preprocessing dokumentu začíná načtením textových dat z dokumentu a pak následuje jejich rozdělení na tokeny. Každý token pak prochází stemmingem, kontrolou na stopWords a na obsažení jiných symbolů než symboly latinské abecedy.

2. Efektivní uložení dokumentů v datové struktuře (invertovaný seznam).

Pokud token(term) projde kontrolou nevýznamnosti - ukládáme ho do tabulky(TermTable), která je reprezentovaná jako mapa. Do této mapy podle klíče(termu) přidáváme do values - index(unikátní) dokumentu, který v tu chvíli zpracováváme.

3. Parsování a vyhodnocení dotazu.

Prvním krokem je extrakce tokenu pro parsování. Parsování pak probíhá s využitím expression/term/factor(ETF) gramatiky, kde se dotaz ukládá do abstraktního syntaktického stromu(AST). Vyhodnocení výrazu je potom rychlejší díky invertovanému seznamu, protože u každého TermNodu je vždycky uložen seznam indexu. Pro vyhodnocení pak zůstává aplikovat logické spojky(AND\NOT\OR) na této seznamy dle struktury výrazu(AST).

Implementace a zdroje

Program se sklada z dvou casti - klientske(Frontend) a serverove(Backend).

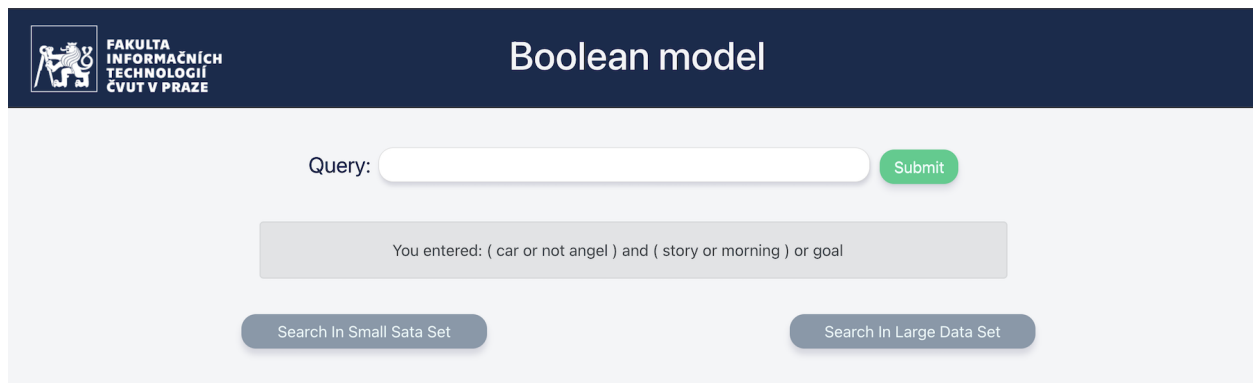
Serverova cast je implimentovana v [Kotlinu](#)(v1.5.13) pomoci frameworku [Ktor](#)(v2.0.0) pro tvorbu a podporu microservis. Za build tool byl zvolen [Maven](#)(4.0.0). Pri implementaci serverove casti byly pouzity nasledujici zdroje a knihovny:

1. Preprocessing.
 - Stemming je implementovany pomoci knihovny [CoreNLP](#).
 - Zdroj seznamu [stopWords](#).
2. Parsovani - [ETF \(expression/term/factor\)](#).
3. DataSety pro zpracovani a vyhledavani:
 - [BBC](#)
 - [Wikipedia](#)
 - [CNN](#)

Webove rozhrani je implementovane za vyuzitim [React.js](#)(v18.0.0) a [Bottstrap](#)(v5.1.3). Komunikace s serverem je zprovoznena pomoci [Axios](#)(v0.27.1). - [react-axios](#)


Příklad výstupu

Vstup: Vstupem může být jakýkoliv validní textový boolean dotaz. Program přijímá dotazy s logickými spojkami (AND/NOT/OR) včetně uzavorkování. Je přidáno kvalitní ošetření -> pokud dotaz je nevalidní vypíše se příslušná chybová hláška.



The screenshot shows a web application titled "Boolean model" with a dark blue header. On the left of the header is the logo of the Faculty of Informatics and Technology at the University of Prague. The main content area has a light gray background. It features a "Query:" label followed by a text input field. To the right of the input field is a green "Submit" button. Below the input field, a gray box displays the entered query: "You entered: (car or not angel) and (story or morning) or goal". At the bottom of the interface, there are two buttons: "Search In Small Data Set" and "Search In Large Data Set".

Do inputu se zadává dotaz. Po kliknutí na tlačítko Submit spustí se kontrola vstupu na Frontendu, která zkontroluje, že dotaz neobsahuje nevhodné symboly a, že vstup není prázdný. Následně se dotaz odesle na server, kde proběhne jeho parsování a syntaktická validace. Pokud je dotaz validní - objeví se label s zadáním dotazu a tlačítka pro možnost vyhledání výrazu v datových kolekcích různé velikosti.



FAKULTA
INFORMAČNÍCH
TECHNOLÓGIÍ
ČVUT V PRAZE

Boolean model

Query:

You entered: (car or not angel) and (story or morning) or goal

Small Data Set Of **200** Files Processed

Sequential Search | Boolean Search

Total found: **51**

Large Data Set Of **2000** Files Processed

Sequential Search | Boolean Search

Total found: **273**

Po kliknutí na tlačítko **Search in Small/Large Data Set** proběhne vyhodnocení výrazu dvěma způsoby popsány v cílech projektu a zobrazí se počet najdených dokumentů.

Tlačítko Show and Compare Search time je pro porovnání dvou způsobů vyhledávání - konkrétně rychlosti vyhledávání. Po kliknutí se zobrazí doba vyhodnocení pro každý způsob, rozdíl v ns a koeficient toho kolikrát rychleji přiběhlo vyhodnocení pomocí invertovaného seznamu.

Tlačítko Print List Of File slouží k zobrazení nalezeného seznamu dokumentů nalezených id.

Viz další stránka.

You entered: (car or not angel) and (story or morning) or goal

Search In Small Sata Set

Search In Large Data Set

Small Data Set Of 200 Files Processed

Sequential Search | Boolean Search

Total found: 51

Show And Compare Search Speed

Search time using boolean model:
231802 ns.

Search tim using sequential search:
453578 ns.

Time difference between first and second:
221776 ns. = 1x

Print List Of Files

Large Data Set Of 2000 Files Processed

Sequential Search | Boolean Search

Total found: 273

Show And Compare Search Speed

Search time using boolean model:
8858244 ns.

Search tim using sequential search:
75084360 ns.

Time difference between first and second:
66226116 ns. = 8x

Print List Of Files

Print List Of Files

fileId: 2
fileId: 8
fileId: 9
fileId: 10
fileId: 14
fileId: 15
fileId: 19
fileId: 21
fileId: 23
fileId: 24
fileId: 26
fileId: 28
fileId: 30
fileId: 31
fileId: 32
fileId: 33
fileId: 34
fileId: 35
fileId: 37
fileId: 38
fileId: 50
fileId: 55
fileId: 56

Print List Of Files

fileId: 2
fileId: 3
fileId: 5
fileId: 6
fileId: 9
fileId: 10
fileId: 11
fileId: 13
fileId: 16
fileId: 17
fileId: 20
fileId: 21
fileId: 24
fileId: 26
fileId: 27
fileId: 28
fileId: 31
fileId: 32
fileId: 33
fileId: 34
fileId: 35
fileId: 36
fileId: 38

Experimentální sekce

Kolekce - 200 dokumentu

query	count	Boolean - ns.	Sequential - ns.	Koef . x
model	37	1557	18741	12
not model	163	78763	804162	10
transfer	14	173448	5008632	20
not transfer	186	155066	2022227	13
Model and transfer	1	881836	5031701	5
Model or transfer	50	9920	2693532	27
Model and not transfer	36	108112	2029255	18
model and transfer or not goal	171	27846	2111287	7

Kolekce 2000 dokumentu

query	count	Boolean - ns.	Sequential - ns.	Koef . x
model	213	34677	2245742	78
not model	1987	87765	2356783	68
transfer	61	86683	1837876	35
not transfer	1939	25008	1361017	54
Model and transfer	13	90063	23026578	55
Model or transfer	61	33316	17303963	59
Model and not transfer	33	23456	6543245	83
model and transfer or not goal	634	79927	4234578	53

Diskuze

Z experimentů je vidět, že čím větší je kolekce tím je časově výhodnější používat invertovaný seznam pro vyhledávání.

Otázkou by mohlo být jak velkou kolekce dat bychom měli mít pro vyhledávání aby se vyplatila implementace měla obrovskou výhodu, protože i když je vidět ten rozdíl v rychlosti pro data jako 200-2000 dokumentů se to stejně vyhledávání nezabírá vůbec čas. Je možné volat API každou vteřinu a proframe se nezasekne. Takže pro zpracování několika tisíc dokumentů implementace by možná byla zbytečná.

Pro obrovské kolekce je to samozřejmě skvělé řešení pro rychlé vyhledávání. Boolean model je současně nekomplikovaný a dost rychle vyhledávání a může ušetřit spoustu času při potřebě zpracovat velké množství dat.

Závěr

1. Boolean model není vhodné implementovat pro malé kolekce dat, ale pro moc velké.
2. Boolean model není komplikovaný v implementaci, ale vyžaduje dodržování velkého množství detailů. Všechny kroky musejí být uделаны bez chyb, protože malá chyba může velmi změnit efektivnost vyhodnocení.
3. Zadání mi přišlo velmi zajímavé. Po dokončení implementace jsem byla schopna provést experimenty s daty a trochu se pohrát s GUI.

Moc se mi libil koncept toho zadani - jako full stack aplikace, protoze je to opravdu dobra prilezitost pochopit proces navrhu programu kdys se na to muzes podivat z obou stran (FE a BE). Prace s daty byla taky zajimava a moc uzitecna. Dekuji.